Data Types and Input Output - I

hitbullseye

Character Set in C

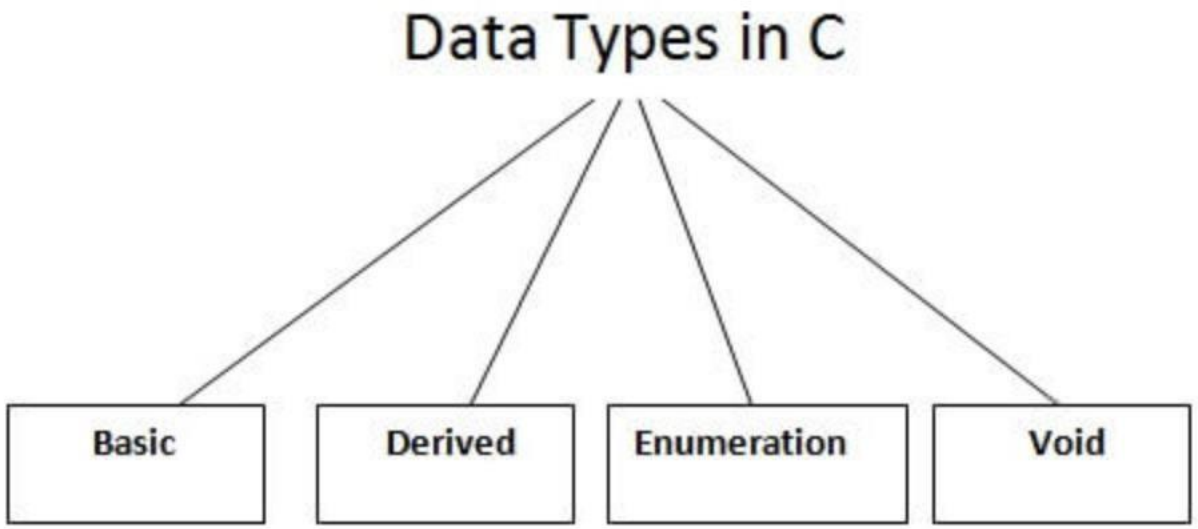
- As every language contains a set of characters used to construct words, statements, etc., C language also has a set of characters which include **alphabets, digits**, and **special symbols**.
- C language supports a total of 256 characters.
- Every C program contains statements. These statements are constructed using words and these words are constructed using characters from C character set. C language character set contains the following set of characters...
  - ✓ Alphabets
  - ✓ Digits
  - ✓ Special Symbols

Character Set in C contd..

- **Alphabets**
  - C language supports all the alphabets from the English language. Lower and upper case letters together support 52 alphabets.
  - lower case letters - **a to z**
  - UPPER CASE LETTERS - **A to Z**
- **Digits**
  - C language supports 10 digits which are used to construct numerical values in C language.
  - Digits - **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**
- **Special Symbols**
  - C language supports a rich set of special symbols that include symbols to perform mathematical operations, to check conditions, white spaces, backspaces, and other special symbols.
  - Special Symbols - **~ @ # $ % ^ & * ( ) _ - + = { } [ ] ; : ' " / ? . > , < \ | tab newline space backspace verticaltab etc.,**

Data Types in C

- Each variable in C has an associated data type.
- Each data type requires different amounts of memory and has some specific operations which can be performed over it.
- The data type is a collection of data with values having fixed values, meaning as well as its characteristics.



Data Types in C

Types	Data Types
Basic Data Type	int, char, float, double
Derived Data Type	array, pointer, structure, union
Enumeration Data Type	enum
Void Data Type	Void

Basic Data Types

- The memory size of the basic data types may change according to 32 or 64-bit operating system.
- The basic data types are also known as the primary data types in C programming.
  - ✓ **Integer –** We use these for storing various whole numbers, such as 5, 8, 67, 2390, etc.
  - ✓ **Character –** It refers to all ASCII character sets. The ASCII character set contains control characters, punctuation marks, digits, and the uppercase and lowercase English alphabets.
  - ✓ **Double –** These include all large types of numeric values that do not come under either floating-point data type or integer data type.
  - ✓ **Floating-point –** These refer to all the real number values or decimal points, such as 40.1, 820.673, 5.9, etc.

Basic Data Types

- C language supports both signed and unsigned types.
- Let's see the basic data types. Its size is given **according to 32-bit architecture.**
- The size of integer may vary from compiler to compiler, while for other data types it remains the same.

Data Types	Memory Size	Range
char	1 byte	-128 to 127
sacred char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
short	2 byte	-32,768 to 32,767
shred short	2 byte	-32,768 to 32,767
unsigned short	2 byte	0 to 65,535
int	2 byte	-32,768 to 32,767
signed int	2 byte	-32,768 to 32.767
unsigned int	2 byte	0 to 65,535
short int	2 byte	-32,768 to 32,767
signed short int	2 byte	-32,768 to 32.767
unsigned short int	2 byte	0 to 65,535
long int	4 byte	-2.147,483,648 to 2,147,483,647
signed long int	4 byte	-2.147,483,648 to 2,147,483,647
unsigned long int	4 byte	0 to 4,294,967,295
float	4 byte	
double	8 byte	
long double	10 byte	

Void Data Type

- **Void –** This term refers to no values at all. We mostly use this data type when defining the functions in a program. We will further discuss the usage of this type in functions.

Keywords used for data types

- Various keywords are used in a program for specifying the data types mentioned above. Here are the keywords that we use:

Keyword Used	Data Type
int	Integer
float	Floating-point
void	Void
char	Character
double	Double

Derived Data Types

- While there are four data types that are primary, various derived data types are also present in C language that help in storing complex types of data.
- The derived data types are basically derived out of the fundamental data types. A derived data type won't typically create a new data type – but would add various new functionalities to the existing ones instead.
- We can derive the derived data types out of the primitive data type by adding some extra relationships to the elements that are available with the primitive data types.
- We use the derived data types to represent multiple values as well as single values in a program.

Types of Derived Data Types in C

- **Arrays –** The *array* basically refers to a sequence (ordered sequence) of a finite number of data items from the same data type sharing one common name.
- **Pointers –** The *Pointers in C* language refer to some special form of variables that one can use for holding other variables' addresses.
- **Unions –** The unions are very similar to the structures. But here, the memory that we allocate to the largest data type gets reused for all the other types present in the group.
- **Structures –** A collection of various different types of data type items that get stored in a contagious type of memory allocation is known as *structure in C*.

Enumerated Types

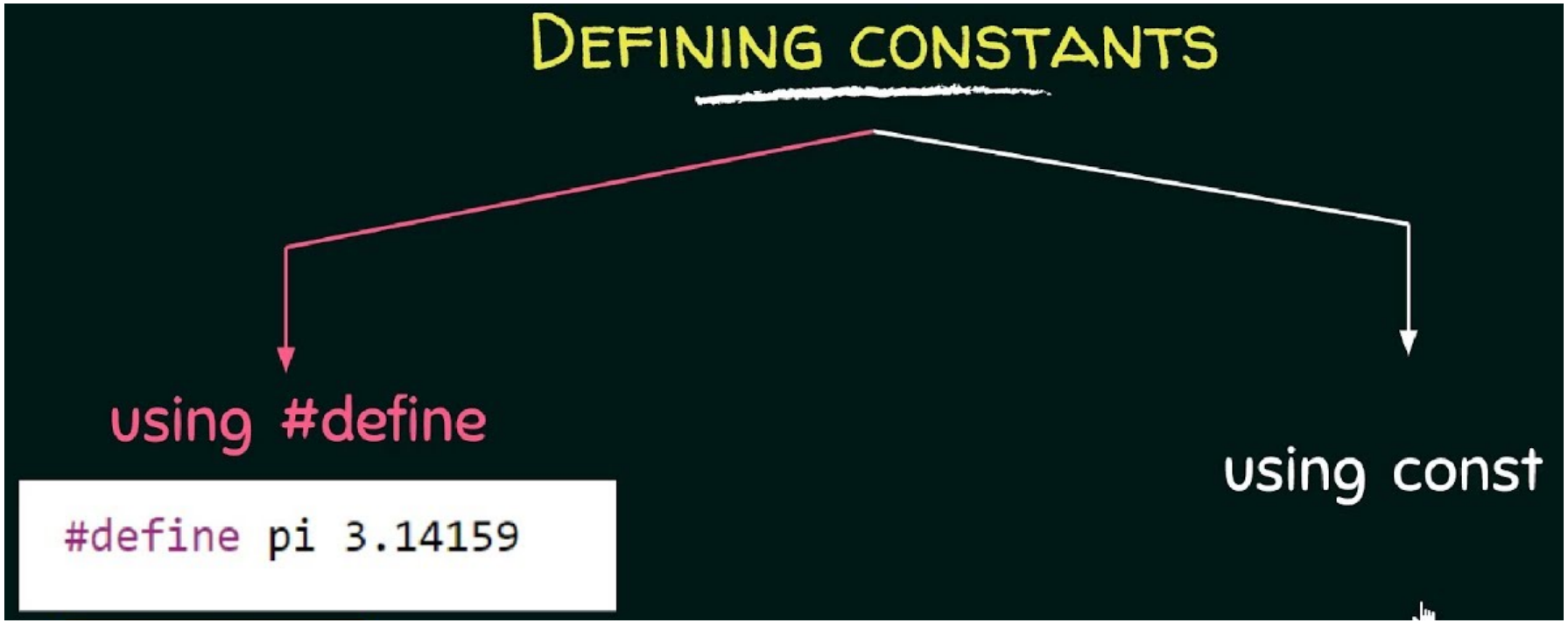
- Enumeration or Enum in C is a special kind of data type defined by the user.
- It consists of constant integrals or integers that are given names by a user.
- The use of enum in C to name the integer values makes the entire program easy to learn, understand, and maintain by the same or even different programmer.
- An enum is defined by using the ‘enum’ keyword in C, and the use of a comma separates the constants within. The basic syntax of defining an enum is:
  - ✓ enum enum_name{int_const1, int_const2, int_const3, …. int_constN};
  - ✓ In the above syntax, the default value of int_const1 is 0, int_const2 is 1, int_const3 is 2, and so on.

Constants

- C identifiers represent the name in the C program, for example, variables, functions, arrays, structures, unions, labels, etc.
- A constant is basically a named memory location in a program that holds a single value throughout the execution of that program.
- It can be of any data type- character, floating-point, string and double, integer, etc.

Creation and Use of Constants in C

- We can create constants in the C programming language by using two of the concepts mentioned below:
  - ✓ By using the '#define' pre-processor
  - ✓ By using the 'const' keyword.



Use of the ‘const’ Keyword

- The ‘const’ keyword is used to create a constant of any given datatype in a program. For creating a constant, we have to prefix the declaration of the variable with the ‘const’ keyword. Here is the general syntax that we follow when using the ‘const’ keyword:
  - ✓ const datatype constantName = value ;OR
  - ✓ const datatype constantName ;
- *Let us look at an example to understand this better*
  - ✓ const int a = 10 ;
- In this case, a is an integer constant that has a fixed value of 10.

```c
#include<stdio.h>

int main() {
    const int var = 67;
    var = 57;
    printf("%d", var);
}
```

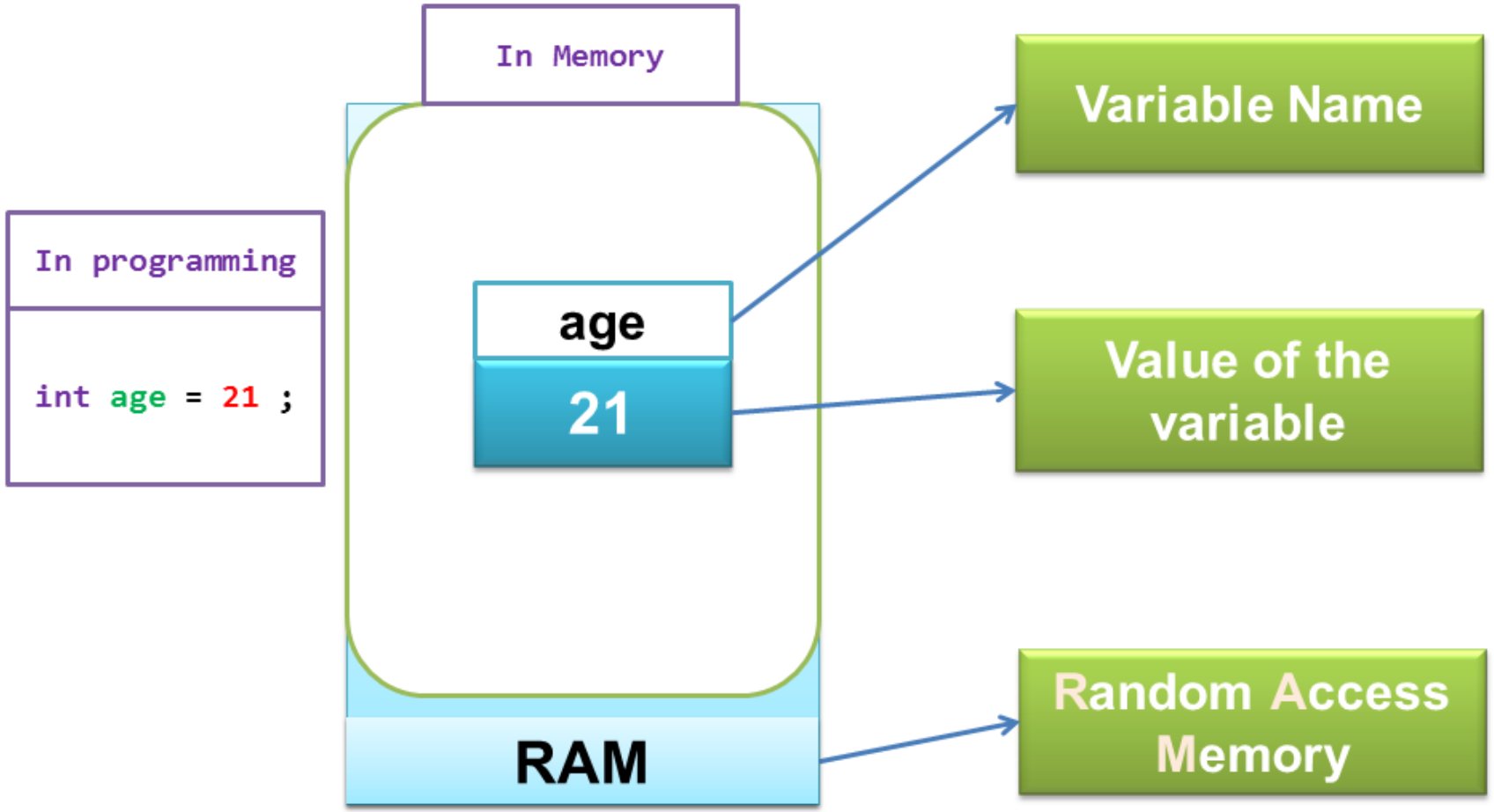
Use of the '#define' preprocessor

- One can also use the '#define' preprocessor directive to create the constants.
- And when we create the constants by making use of the preprocessor directive, we must define it in the very beginning of the program.
- Here is the syntax that we must use for creating a constant by making use of the '#define' preprocessor directive:
  - ✓ #define CONSTANTNAME value
  - ✓ In program PI is also called as macro

```c
#include<stdio.h>
#define PI 3.14159
int main() {
    printf("%.5f", PI);
    return 0;
}
```

Variables

- Variable is used to store the value. As name indicates its value can be changed or also it can be reused many times.



Types of Variables in C

- **Local variable –** A variable which is declared inside the function is known as local variable. It is used only inside the function in which it is declared.
- **Global variable –** A variable which is declared outside the function is known as global variable. It can be used throughout the program.
- **Static variable –** It is used to retain its value between multiple function calls. It is declared using static keyword.
- **External variable –** You can share a variable in multiple C source files by using external variable. It is declared using extern keyword.
- **Automatic variable –** Variable which is declared inside the block is known as automatic variable by default.

Variable Naming

- Variables in C may be given representations containing multiple characters. But there are rules for these representations.
- Variable names in C
  - ✓ May only consist of letters, digits, and underscores
  - ✓ May be as long as you like, but only the first 31 characters are significant
  - ✓ May not begin with a number
  - ✓ May not be a C **reserved word (keyword)**

Variable Declaration

Variable Declaration

- Before using a variable, you must give the compiler some information about the variable; i.e., you must **declare** it.
- The **declaration statement** includes the **data type** of the variable.
- Examples of variable declarations:

```
int flowers;
float  area ;
```

- When we declare a variable
  - Space is set aside in memory to hold a value of the specified data type
  - That space is associated with the variable name
  - That space is associated with a unique **address**
- Visualization of the declaration

```
int meatballs ;
```

Keywords in C

- A keyword is a **reserved word**. You cannot use it as a variable name, constant name, etc. There are only 32 reserved words (keywords) in the C language.
- A list of 32 keywords in the c language is given below:

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

Identifiers in C

- C identifiers represent the name in the C program, for example, variables, functions, arrays, structures, unions, labels, etc.
- An identifier can be composed of letters such as uppercase, lowercase letters, underscore, digits, but the starting letter should be either an alphabet or an underscore.
- We can say that an identifier is a collection of alphanumeric characters that begins either with an alphabetical character or an underscore, which are used to represent various programming elements such as variables, functions, arrays, structures, unions, labels, etc.

Rules for constructing C identifiers

- The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digit, or underscore.
- It should not begin with any numerical digit.
- In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
- Commas or blank spaces cannot be specified within an identifier.
- Keywords cannot be represented as an identifier.
- The length of the identifiers should not be more than 31 characters.
- Identifiers should be written in such a way that it is meaningful, short, and easy to read.

Keywords Vs Identifiers

Keyword	Identifier
Keyword is a pre-defined word.	The identifier is a used-defined word
It must be written in a lowercase letter.	It can be written in both lowercase and uppercase letters.
Its meaning is pre-defined in the c compiler.	Its meaning is not defined in the c compiler.
It is a combination of alphabetical characters.	It is combination of alphanumeric characters.
It does not contain the underscore character.	It can contain the underscore character.

References Link

- https://www.w3schools.com/c/index.php
- https://archive.nptel.ac.in/courses/106/104/106104128/
- https://medium.com/@eitworld/c-data-type-5ab5a0773f17

THANK YOU

hitbullseye